Writer Identification Using Inexpensive Signal Processing Techniques

Serguei A. Mokhov
Computer Science
and Software Engineering,
Concordia University, Montreal, Canada,
Email: `mokhov@cse.concordia.ca`

Miao Song
Graduate School,
Concordia University,
Montreal, Canada,
Email: `m_song@cse.concordia.ca`

Ching Y. Suen
Centre for Pattern Recognition
and Machine Intelligence,
Concordia University, Montreal, Canada,
Email: `suen@cenparmi.concordia.ca`

Abstract

We propose to use novel and classical audio and text signal-processing and otherwise techniques for "inexpensive" fast writer identification tasks of scanned hand-written documents "visually". The "inexpensive" refers to the efficiency of the identification process in terms of CPU cycles while preserving decent accuracy for preliminary identification. This is a comparative study of multiple algorithm combinations in a pattern recognition pipeline implemented in Java around an open-source Modular Audio Recognition Framework (MARF) that can do a lot more beyond audio. We present our preliminary experimental findings in such an identification task. We simulate "visual" identification by "looking" at the hand-written document as a whole rather than trying to extract fine-grained features out of it prior classification.

Keywords: writer identification, Modular Audio Recognition Framework (MARF), signal processing, simulation

1 Introduction

1.1 Problem Statement

Current techniques for writer identification often rely on the classical tools, methodologies, and algorithms in handwriting recognition (and in general in any image-based pattern recognition) such as skeletonizing, contouring, line-based and angle-based feature extraction, and many others. Then those techniques compare the features to the "style" of features a given writer may have in the trained database of known writers. These classical techniques are, while highly accurate, are also time consuming for bulk processing of a large volume of digital data of handwritten material for its preliminary or secondary identification of who may have written what.

1.2 Proposed Solution

We simulate "quick visual identification" of the hand-writing of the writer by looking at a page of hand-written text as a whole to speed up the process of identification, especially when one needs to do a quick preliminary classification of a large volume of documents. For that we treat the sample pages as either 1D or 2D arrays of data and apply 1D or 2D loading using various loading methods, then 1D or 2D filtering, then in the case of 2D filtering, we flatten a 2D array into 1D prior feature extraction, and then we continue the classical feature extraction, training and classification tasks using a comprehensive algorithm set within Modular Audio Recognition Framework (MARF)'s implementation, by roughly treating each hand-written image sample as a wave form as in e.g. in speaker identification. We insist on

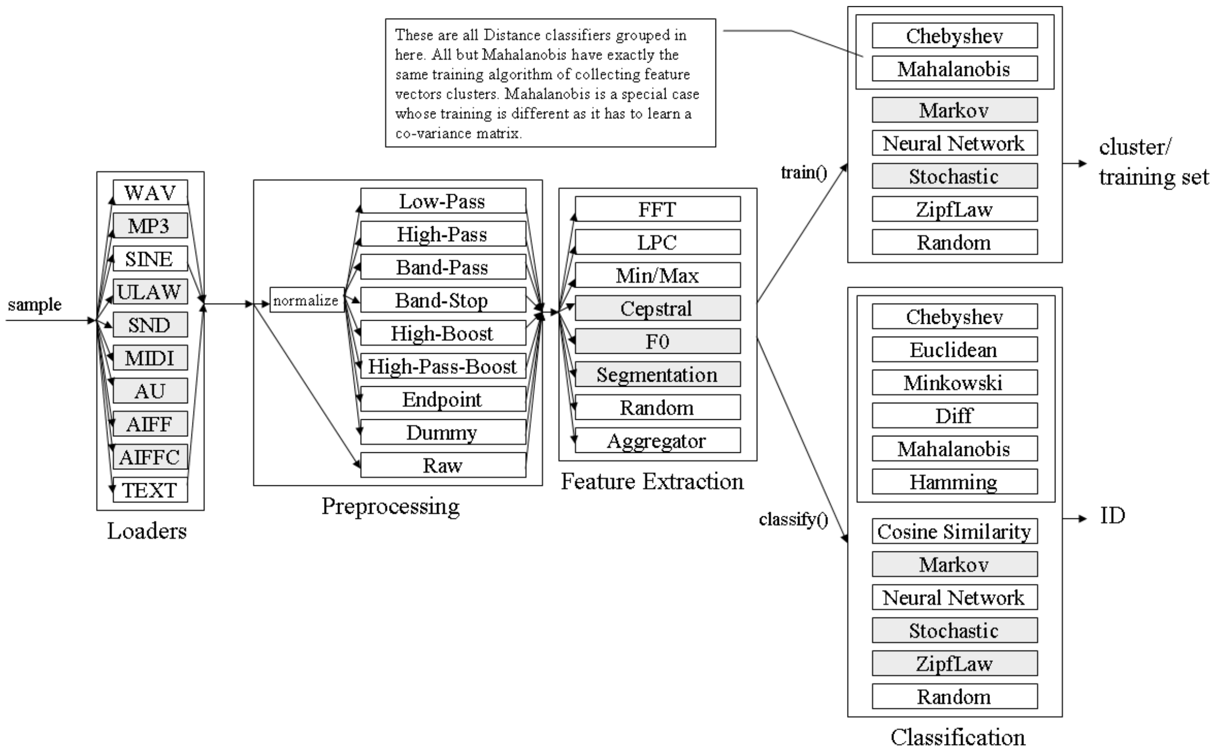

Figure 1: MARF's Pattern Recognition Pipeline

1D as it is the baseline storage mechanism for MARF and it is less storage consuming while sufficient to achieve high accuracy in the writer identification task.

This approach is in a way similar to the one where MARF was applied to file type analysis for forensic purposes [20] using machine learning and assuming each file is a sort of a signal on Unix systems as compared to the traditional `file` utility [3, 4].

1.3 Introduction to MARF

Modular Audio Recognition Framework (MARF) is an open-source collection of pattern recognition APIs and their implementation for unsupervised and supervised machine learning and classification written in Java [13, 24, 17, 15, 16, 18]. One of its design purposes is to act as a testbed to try out common and novel algorithms found in literature and industry for sample loading, preprocessing, feature extraction, and training and classification tasks. One of the main goals and design approaches of MARF is to provide scientists with a tool for comparison of the algorithms in a homogeneous environment and allowing the dynamic module selection (from the implemented modules) based on the configuration options supplied by applications. Over the course of several years MARF accumulated a fair number of implementations for each of the pipeline stages allowing reasonably comprehensive comparative studies of the algorithms *combinations*, and studying their *combined* behavior and other properties when used for various pattern recognition tasks. MARF is also designed to be very configurable while keeping the generality and some sane default settings to "run-off-the-shelf" well. MARF and its derivatives, and applications were also used beyond audio processing tasks due to the generality of the design and implementation in [14, 21, 19, 22, 23] and other unpublished or in-progress works.

1.3.1 Classical Pattern Recognition Pipeline

The conceptual pattern recognition pipeline shown in Figure 1 depicts the core of the data flow and transformation between the stages of the MARF's pipeline. The inner boxes represent most of the available concrete module implementations or stubs. The grayed-out boxes are either the stubs or partly implemented. The white boxes signify implemented algorithms. Generally, the whole pattern recognition process starts by loading a sample (e.g. an audio recording in a wave form, a text, or image file), preprocessing it (removing noisy and "silent" data and other unwanted elements), then extracting the most prominent features, and finally either training of the system such that the system either learns a new set of a features of a given subject or actually classify and identify what/how the subject is. The outcome of training is either a collection of some form of feature vectors or their mean or median clusters, which a stored per subject learned. The outcome of classification is a 32-bit unique integer usually indicating who/what the subject the system believes is. MARF designed to be a standalone `marf.jar` file required to be usable and has no dependencies on other libraries. Optionally, there is a dependency for debug versions of `marf.jar` when JUnit [5] is used for unit testing.

1.3.2 Algorithms

MARF has actual implementations of the framework's API in a number of algorithms to demonstrate its abilities in various pipeline stages and modules. There are a number of modules that are under the process of implementation or porting from other project for comparative studies that did not make it to this work at the time of its writing. Thus, the below is an incomplete summary of implemented algorithms corresponding to the Figure 1 with a very brief description:

- Fast Fourier transform (FFT), used in FFT-based filtering as well as feature extraction [2].
- Linear predictive coding (LPC) used in feature extraction.
- Artificial neural network (classification).
- Various distance classifiers (Chebyshev, Euclidean, Minkowski [1], Mahalanobis [12], Diff (internally developed within the project, roughly similar in behavior to the UNIX/Linux `diff` utility [11]), and Hamming [7]).
- Cosine similarity measure [6, 10], which was thoroughly discussed in [9] and often produces the best accuracy in this work in many configurations (see further).
- Zipf's Law-based classifier [25].
- General probability classifier.
- Continuous Fraction Expansion (CFE)-based filters [8].
- A number of math-related tools, for matrix and vector processing, including complex numbers matrix and vector operations, and statistical estimators used in smoothing of sparse matrices (e.g. in probabilistic matrices or Mahalanobis distance's covariance matrix). All these are needed for MARF to be self-contained.

2 Methodology

To enable the experiments in this work and their results we required to do the alteration of the MARF's pipeline through its plug-in architecture. We outline the modifications and the experiments and conducted using a variety of options.

2.1 Modified MARF's Pipeline

There are slight modifications to the pipeline that were required to MARF's original pipeline in order to enable some of the experiments outlined below for the writer identification tasks. Luckily, due to MARF's extensible architecture we can do those modifications as plug-ins, primarily for sample loading and preprocessing, that we plan on integrating into the core of MARF.

2.1.1 Loaders

We experiment with a diverse scanned image sample loading mechanisms to see which contribute more to the better accuracy results and the most efficient. There is a naive and less naive approach to do so. We can treat the incoming sample as:

- an image, essentially a 2D array, naturally
- a byte stream, i.e. just a 1D array of raw bytes
- a text file, treat the incoming bytes as text, also 1D
- a wave form, as if it is encoded WAVE file, also 1D

Internally, regardless the initial interpretation of the scanned hand-written image samples, the data is always treated as some wave form or another. The initial loading affects the outcome significantly, and we tried to experiment which one yields better results, which we present as options. For this to work we had to design and implement `ImageSampleLoader` as an external to MARF plug-in to properly decode the image data as image and return a 2D array representation of it (it is later converted to 1D for further processing). We adapt the `ImageSample` and `ImageSampleLoader` previously designed for the `TestFilters` application of MARF for 2D filter tests. The other loaders were already available in MARF's implementation, but had to be subclassed or wrapped around to override some settings. Specifically, we have a `ByteArrayFileReader`, `TextLoader`, and `WAVLoader` in the core MARF that we rely upon. For the former, since it does not directly implement the `ISampleLoader` interface, we also create an external wrapper plug-in, `RawSampleLoader`. The `TextLoader` provides options for loading the data as uni-, bi-, and tri-gram models, i.e. one sample point consists of one, two, or three characters. The `WAVLoader` allows treating the incoming sample at different sample rates as well, e.g. 8000 kHz, 16000 kHz, and so on. We had to create a `TIFFtoWAVLoader` plug-in for this research work to allow the treatment of the TIFF files as WAV with the proper format settings.

2.1.2 Filters

The Filter Framework of MARF and its API represented by the `IFilter` interface has to be invoked with the 2D versions of the filters instead of 1D, which is a sufficient default for audio signal processing. The Filter framework has a 2D API processing that can be applied to images, "line-by-line". The 2D API of `IFilter` returns a 2D results. In order for it to be usable by the rest of the pipeline, it has to be "flattened" into a 1D array. The "flattening" can be done row-by-row or column-by-column; we experiment with both ways of doing it. Once flattened, the rest of the pipeline process functions as normal. Since there is no such a default preprocessing module in the core MARF, we implement it as a preprocessing plug-in in this work, which we call `Filter2Dto1D`. This class implements `preprocess()` method to behave the described way. This class in itself actually does not do much, but instead the FFT-based set of filters is mirrored from the core MARF to this plug-in to adhere to this new implementation of `preprocess()` and at the same time to delegate all the work to the core modules. Thus, we have the base `FFTFilter2Dto1D`, and the concrete `LowPass2Dto1D`, `HighPass2Dto1D`, `BandStop2Dto1D`, and `BandPass2Dto1D` FFT-based filters. The CFE-based filters require further testing at this point and as such were not included in the experiments.

2.1.3 Noise Removal

We employ two basic methodologies of noise removal in our experiments: (1) we either remove the noise by loading the "noise" sample, a scanned "blank" sheet with no writings on it. Subtracting the frequencies of this noise sample from the incoming samples gives us the net effect of large noise removal. This FFT sample-based noise remover is only effective for the 2D preprocessing operations. Implementation-wise, we implement it in the `SampleBasedNoiseRemover` preprocessing plug-in class. (2) We compare that to the default noise removal in MARF that is constructed by application of the plain 1D low-pass FFT filter.

2.2 `WriterIdentApp`

We provide a testing application, called `WriterIdentApp`, to do all the experiments in this work and statistics gathering. The application is a writer-identification-oriented fork of `SpeakerIdentApp` present within MARF's repository [24] for speaker identification task. The application has been amended with the options to accept the four loader types instead of two, noise removal by subtraction of the noise sample, and the 2D filtering plug-ins. The rest of the application options are roughly the same as that of `SpeakerIdentApp`. Like all of MARF and its applications, `WriterIdentApp` will be released as open-source and can be made available to the willing upon request prior that.

2.3 Resolution

We vary the resolution of our samples as 600dpi, 300dpi, and 96dpi in our experiments to see how it affects the accuracy of the identification. The samples are both grayscale and black-and-white.

3 Testing, Experiments, and Results

The handwritten training samples included two pages scanned from students' quizzes. The testing performed on the another, third page of the same exam for each student. The total number of students's exams in class studied is 25.

3.1 Setup

In the setup we are testing multiple permutation of configurable parameters, which are outlined below.

3.1.1 Samples

The samples are scanned pages letter-sized as uncompressed TIFF images of the following resolutions and color schemes:

- 600 dpi grayscale, black-and-white
- 300 dpi grayscale, black-and-white
- 96 dpi grayscale, black-and-white

3.1.2 Sample Loaders

- Text loader: unigram, bigram, trigram
- WAVE loader: PCM, 8000 kHz, mono, 2 bytes per amplitude sample point
- Raw loader: byte loader (1-byte, 2-byte, 4-byte per sample point)
- TIFF Image 2D loader

Byte loader and text loader are similar but not identical. In Java characters are in UNICODE and occupy physically two bytes and we use a character-oriented reader to do so. In the byte loader, we deal with the raw bytes and our "ngrams" correspond to the powers of 2.

3.1.3 Preprocessing

- 1D filtering works with 1D loaders, and low-pass FFT filter acts as a noise remover
- 2D filtering covered by a plug-in with 2D FFT filters and noise sample subtraction
- Flattening of the 2D data to 1D by row or column

3.1.4 Feature Extraction and Classification

The principle fastest players in the experimentation so-far were primarily the distance and similarity measure classifiers and for feature extraction FFT, LPC and min/max amplitudes. All these modules are at their defaults as defined by MARF [24, 15, 17, 16, 18].

3.2 Analysis

Generally, it appears the 2D versions of the combinations produce higher accuracy. The text-based and byte-based loaders perform at the average level and the wave form loading slightly better. The black-and-white images at all resolutions obviously load faster as they are much smaller in size, and even the 96 dpi-based image performed very well suggesting the samples need not be of the highest quality. Algorithm combinations that had silence removed after either 1D or 2D based noise removal contributed to the best results by eliminating "silence gaps" (in the image strings of zeros, similar to compression) thereby making samples looking quite distinct. The noise-sample based removal, even eliminates the printed text and lines of the handed-out exam sheets keeping only hand-written text and combined with the silence removal pushes the accuracy percentage even higher.

The experiments are running on two major hardware pieces: a Dell desktop and a server with two 2 CPUs. Run-time that it takes to train the system on 50 600dpi grayscale samples (35Mb each) is varied between 15 to 20 minutes on a Dell Precision 370 workstation with 1GB of RAM and 1.5GHz Intel Pentium 4 processor running Fedora Core 4 Linux. For the testing samples, it takes between 4 and 7 seconds depending on the algorithm combination for the 35Mb testing samples. All the sample files were read off a DVD disk, so the performance was less optimal than from a hard disk. In the case of the server with two Intel Pentium 4 CPUs, 4GB of ram and the four processing running it takes 2-3 times faster for the same amount. 96dpi b/w images take very fast to process and offer the best response times.

4 Conclusion

As of this writing due two numerous exhaustive combinations and about 600 runs per loader most of the experiments and some testing are still underway and are expected to complete within a week. The pages with the list of resulting tables are obviously not fitting within a 6-page conference paper, but will be made available in full upon request. Some of the fastest results have come back entirely, but for now they show disappointing accuracy performance of 20% correctly identified writiers in our settings, which is way below expected from our hypothesis. Since the results are incomplete, the authors are reviewing them as they come in and seek the faults in the implementation and data. The complete set of positive or negative outcomes will be summarized in the final version of the article.

4.1 Applications

We outline possible applications of our quick classification approach:

- Students' exams verification in case of fraud claims to quickly sort out the pages into appropriate bins
- For large amount of scanned checks (e.g. same as banks make available on-line). Personal checks identification can be used to see if the system can tell they are written by the same person. In this case the author used his personal check scans due to their handy availability.
- Quick sorting out of hand-written mail.
- Blackmail investigation when checking whether some letters with threats were written by the same person or who that person might be by taking sample handwriting samples of the suspects in custody or investigation.

4.2 Future Work

- Further improve recognition accuracy by investigating more algorithms and their properties.
- Experiment with the CFE-based filters.
- Automation of training process for the sorting purposes.
- Export results in Forensic Lucid for forensic analysis.

4.3 Improving Identification Accuracy

So far, we did a quick way of doing writer authentication without using any common advanced or otherwise image processing techniques, such as contouring, skeletonizing, etc. and the related feature extraction, such as angles, lines, direction, relative position of them, etc. We can "inject" those approaches into the available pipeline if we can live with slower processing speeds due to the additional overhead induced by the algorithms, but improve the accuracy of the identification.

Acknowledgments

This work is partially funded by NSERC, FQRSC, and Graduate School and the Faculty of Engineering and Computer Science, Concordia University, Montreal, Canada.

References

[1] H. Abdi. Distance. In N.J. Salkind, editor, *Encyclopedia of Measurement and Statistics*, Thousand Oaks (CA): Sage, 2007.

[2] Stefan M. Bernsee. *The DFT "à pied": Mastering The Fourier Transform in One Day*. DSPdimension.com, 1999-2005. `http://www.dspdimension.com/data/html/dftapied.html`.

[3] Ian F. Darwin, John Gilmore, Geoff Collyer, Rob McMahon, Guy Harris, Christos Zoulas, Chris Lowth, Eric Fischer, and Various Contributors. `file` – determine file type, BSD General Commands Manual, file(1). BSD, January 1973–2007. man file(1).

[4] Ian F. Darwin, John Gilmore, Geoff Collyer, Rob McMahon, Guy Harris, Christos Zoulas, Chris Lowth, Eric Fischer, and Various Contributors. `file` – determine file type. [online], March 1973–2008. `ftp://ftp.astron.com/pub/file/`, last viewed April 2008.

[5] Erich Gamma and Kent Beck. JUnit. Object Mentor, Inc., 2001–2004. `http://junit.org/`.

[6] E. Garcia. Cosine similarity and term weight tutorial, 2006. `http://www.miislita.com/information-retrieval-tutorial/cosine-similarity-tutorial.html`.

[7] Richard W. Hamming. Error Detecting and Error Correcting Codes. Bell System Technical Journal 26(2):147-160, 1950. `http://en.wikipedia.org/wiki/Hamming_distance`.

[8] Shivani Haridas. Generation of 2-D digital filters with variable magnitude characteristics starting from a particular type of 2-variable continued fraction expansion. Master's thesis, Concordia University, Montréal, Canada, July 2006.

[9] Michelle Khalifé. Examining orthogonal concepts-based micro-classifiers and their correlations with noun-phrase coreference chains. Master's thesis, Concordia University, Montréal, Canada, 2004.

[10] Anand Kishore. Similarity measure: Cosine similarity or euclidean distance or both, February 2007. `http://semanticvoid.com/blog/2007/02/23/similarity-measure-cosine-similarity-or-euclidean-distance-or-both/`.

[11] D. Mackenzie, P. Eggert, and R. Stallman. Comparing and merging files. [online], 2002. `http://www.gnu.org/software/diffutils/manual/ps/diff.ps.gz`.

[12] P.C. Mahalanobis. On the generalised distance in statistics. Proceedings of the National Institute of Science of India 12 (1936) 49-55, 1936. `http://en.wikipedia.org/wiki/Mahalanobis_distance`.

[13] Serguei Mokhov, Ian Clement, Stephen Sinclair, and Dimitrios Nicolacopoulos. Modular Audio Recognition Framework. Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, 2002–2003. Project report, `http://marf.sf.net`, last viewed April 2008.

[14] Serguei A. Mokhov. On design and implementation of distributed modular audio recognition framework: Requirements and specification design document. [online], August 2006. Project report, `http://arxiv.org/abs/0905.2459`, last viewed May 2009.

[15] Serguei A. Mokhov. Introducing MARF: a modular audio recognition framework and its applications for scientific and software engineering research. In *Advances in Computer and Information Sciences and Engineering*, pages 473–478, University of Bridgeport, U.S.A., December 2007. Springer Netherlands. Proceedings of CISSE/SCSS'07.

[16] Serguei A. Mokhov. Choosing best algorithm combinations for speech processing tasks in machine learning using MARF. In Sabine Bergler, editor, *Proceedings of the 21st Canadian AI'08*, pages 216–221, Windsor, Ontario, Canada, May 2008. Springer-Verlag, Berlin Heidelberg. LNAI 5032.

[17] Serguei A. Mokhov. Experimental results and statistics in the implementation of the modular audio recognition framework's API for text-independent speaker identification. In C. Dale Zinn, Hsing-Wei Chu, Michael Savoie, Jose Ferrer, and Ante Munitic, editors, *Proceedings of the 6th International Conference on Computing, Communications and Control Technologies (CCCT'08)*, volume II, pages 267–272, Orlando, Florida, USA, June 2008. IIIS.

[18] Serguei A. Mokhov. Study of best algorithm combinations for speech processing tasks in machine learning using median vs. mean clusters in MARF. In Bipin C. Desai, editor, *Proceedings of C3S2E'08*, pages 29–43, Montreal, Quebec, Canada, May 2008. ACM. ISBN 978-1-60558-101-9.

[19] Serguei A. Mokhov. Towards security hardening of scientific distributed demand-driven and pipelined com-

puting systems. In *Proceedings of the 7th International Symposium on Parallel and Distributed Computing (ISPDC'08)*, pages 375–382, Krakow, Poland, July 2008. IEEE Computer Society.

[20] Serguei A. Mokhov and Mourad Debbabi. File type analysis using signal processing techniques and machine learning vs. `file` unix utility for forensic analysis. In Oliver Goebel, Sandra Frings, Detlef Guenther, Jens Nedon, and Dirk Schadt, editors, *Proceedings of the IT Incident Management and IT Forensics (IMF'08)*, pages 73–85, Mannheim, Germany, September 2008. GI. LNI140.

[21] Serguei A. Mokhov, Lee Wei Huynh, and Jian Li. Managing distributed MARF with SNMP. Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Canada, April 2007. Project Report. Hosted at `http://marf.sf.net`, last viewed April 2008.

[22] Serguei A. Mokhov, Lee Wei Huynh, Jian Li, and Farid Rassai. A privacy framework within the java data security framework (JDSF): Design refinement, implementation, and statistics. In Nagib Callaos, William Lesso, C. Dale Zinn, Jorge Baralt, Jaouad Boukachour, Christopher White, Thilidzi Marwala, and Fulufhelo V. Nelwamondo, editors, *Proceedings of the 12th World Multi-Conference on Systemics, Cybernetics and Informatics (WM-SCI'08)*, volume V, pages 131–136, Orlando, Florida, USA, June 2008. IIIS.

[23] Serguei A. Mokhov, Lingyu Wang, and Jian Li. Simple dynamic key management in SQL randomization. Unpublished, 2009.

[24] The MARF Research and Development Group. The Modular Audio Recognition Framework and its Applications. [online], 2002–2009. `http://marf.sf.net`, last viewed October 2009.

[25] George Kingsley Zipf. *The Psychobiology of Language*. Houghton-Mifflin, New York, NY, 1935. `http://en.wikipedia.org/wiki/Zipf%27s_law`.